IN THE CLAIMS

Claims 1-37. (Canceled)

38. (New) A method for testing remote target applications, said method comprising the steps of:

obtaining meta-information of a target application;

comparing the obtained meta-information with pre-stored meta-information;

updating the pre-stored meta-information when a discrepancy between the obtained meta-information and the pre-stored meta-information is detected;

automatically generating test cases based on the obtained meta-information;

automatically creating test scenarios;

generating the test cases from the test scenarios;

automatically generating test programs using the test scenarios and the test cases;

building a test image from the test programs;

downloading the test image to the target application for testing;

automatically testing the target application;

generating reports from test results in a desired format;

providing a framework to define the test scenarios by using the obtained meta-information;

automatically generating different test cases using the test scenarios; and

generating the test programs in a description language using the test scenarios and the test cases.

39. (New) The method as claimed in claim 38, wherein

the meta-information of the target application are obtained by using a reflection principle in one of two ways; (i) by utilizing a reflection object bundled with the target application or (ii) by downloading the reflection object to the target application.

40. (New) The method as claimed in claim 38, wherein

the test scenarios, the test programs and the test image are generated by utilizing object serialization in order to improve data communication security over a network, as well as to improve utilization of resources in the network in order to reduce time of execution.

41. (New) The method as claimed in claim 38, wherein

the test programs are generated independently of the Application Programming Interfaces (APIs).

42. (New) The method as claimed in claim 38, wherein

execution of the test programs is conducted by a user utilizing an order of execution, a repetition, a requirement for resetting and batch information.

43. (New) The method as claimed in claim 38, wherein

the reports are generated for each specified test scenario.

44. (New) The method as claimed in claim 38, wherein

a solution is provided to a service station for testing the target application or the service station utilizes an automatic test system through a terminal provided at the service station.

45. (New) The method as claimed in claim 38, wherein

a plurality of target applications are simultaneously tested either at one location or at multiple locations.

46. (New) The method as claimed in claim 38, wherein

the framework is a Boolean foo function which results in regular testing and irregular testing.

47. (New) The method as claimed in claim 46, wherein

the regular testing is a Boundary Value Analysis (BVA) technique that includes a parameter being an integer type having a range between 0 and 100.

48. (New) The method as claimed in claim 46, wherein

the regular testing is a Boundary Value Analysis (BVA) technique that includes a parameter being a float type having a range between 0 and 1000.

49. (New) The method as claimed in claim 46, wherein

the regular testing is a Equivalence Partitioning (EP) technique that includes a Boolean having only two test case values.

50. (New) The method as claimed in claim 38, wherein

resetting is performed when a user determines that the test programs contain execution errors.

51. (New) The method as claimed in claim 38, wherein

the remote target applications are identified with an Internet Protocol (IP) address.

52. (New) An automatic test system for testing remote target applications, said system comprising:

obtaining means for obtaining meta-information of a target application;

comparing means for comparing the obtained meta-information with pre-stored meta-information stored in a storage means;

updating means for updating the pre-stored meta-information when a discrepancy between the obtained meta-information and the pre-stored meta-information is detected;

first generating means for automatically generating test cases based on the obtained meta-information;

test scenario creating means for automatically creating test scenarios;

second generating means for generating the test cases from the test scenarios;

third generating means for automatically generating test programs using the test scenarios and the test cases;

image builder means for building a test image from the test programs;

downloading means for downloading the test image to the target application for testing;

testing means for automatically testing the target application;

fourth generating means for generating reports from test results in a desired format;

providing means for providing a framework to define the test scenarios by using the obtained meta-information;

fifth generating means for automatically generating different test cases using the test scenarios; and

sixth generating means for generating the test programs in a description language using the test scenarios and the test cases.

53. (New) The automatic test system as claimed in claim 52, wherein

the meta-information of the target application is obtained by using a reflection principle by utilizing a reflection object bundled with the target application.

54. (New) The automatic test system as claimed in claim 52, wherein

the meta-information of the target application is obtained by using a reflection principle by downloading a reflection object to the target application.

55. (New) The automatic test system as claimed in claim 52, wherein

the test scenarios, the test programs and the test image are generated by utilizing object serialization in order to improve data communication security over a network, as well as to improve utilization of resources in the network in order to reduce time of execution.

56. (New) The automatic test system as claimed in claim 52, wherein

the test programs are generated independently of the Application Programming Interfaces (APIs).

57. (New) The automatic test system as claimed in claim 52, wherein

execution of the test programs is conducted by a user utilizing an order of execution, a repetition, a requirement for resetting and batch information.

58. (New) The automatic test system as claimed in claim 52, wherein

the reports are generated for each specified test scenario.

59. (New) The automatic test system as claimed in claim 52, wherein

a solution is provided to a service station for testing the target application or the service station utilizes an automatic test system through a terminal provided at the service station.

60. (New) The automatic test system as claimed in claim 52, wherein

a plurality of target applications are simultaneously tested either at one location or at multiple locations.

61. (New) The automatic test system as claimed in claim 52, wherein

the framework is a Boolean foo function which results in regular testing and irregular testing.

62. (New) The automatic test system as claimed in claim 61, wherein

the regular testing is a Boundary Value Analysis (BVA) technique that includes a parameter being an integer type having a range between 0 and 100.

63. (New) The automatic test system as claimed in claim 61, wherein

the regular testing is a Boundary Value Analysis (BVA) technique that includes a parameter being a float type having a range between 0 and 1000.

64. (New) The automatic test system as claimed in claim 61, wherein

the regular testing is a Equivalence Partitioning (EP) technique that includes a Boolean having only two test case values.

65. (New) The automatic test system as claimed in claim 52, wherein

resetting is performed when a user determines that the test programs contain execution errors.

66. (New) The automatic test system as claimed in claim 52, wherein

the remote target applications are identified with an Internet Protocol (IP) address.

67. (New) The automatic test system as claimed in claim 52, wherein

the first generating means further comprises a configuration module, a test design module, a test driver module, a test execution module and a report module that are connected to the storage means via a network.

68. (New) The automatic test system as claimed in claim 67, wherein

the configuration module is a software program executed on a computing system that obtains information on test techniques, objects and data types from a user for defining the test cases.

69. (New) The automatic test system as claimed in claim 67, wherein

the test design module is a software program executed on a computing system that provides the framework of the test scenarios.

70. (New) The automatic test system as claimed in claim 67, wherein

the test driver module is a software program executed on a computing system that automatically generates the test cases and the test programs in the description language by utilizing the test scenarios provided by the test design module.

71. (New) The automatic test system as claimed in claim 67, wherein

the execution module loads the test image created by an image builder module on the target application, and monitors and controls the execution of the test image of the target application.

72. (New) The automatic test system as claimed in claim 52, wherein

the image builder means converts the test programs received from the test driver module in the description language to an image form suitable for loading and executing on the target application.

73. (New) The automatic test system as claimed in claim 67, wherein

the report module generates reports from the results of the testing on the target application, which are stored in the data storage means.

74. (New) The automatic test system as claimed in claim 52, wherein

the data storage means stores information relating to a test scenario, a test technique, an object, results of tests and incorporates object serialization means in order to improve execution time and security.

75. (New) The automatic test system as claimed in claim 52, wherein

the testing means is developed in a programming language that is hardware and software independent.

76. (New) The automatic test system as claimed in claim 52, wherein

the description language is Standard Description Language (SDL).

77. (New) The automatic test system as claimed in claim 52, wherein

the description language is converted by a language code converter to a desired test language.

78. (New) The automatic test system as claimed in claim 77, wherein

the language code converter converts a description language test program to a desired language test program that is provided either at the test driver module of the test generation means or at the image builder means.

79. (New) The automatic test system as claimed in claim 52, wherein

the data storage means is a server and is not dependent on any particular database, the server being developed in a programming language that is hardware and software independent.

80. (New) The automatic test system as claimed in claim 52, wherein

the image builder means comprises a compiler and a linker in order to generate an executable data image.

81. (New) The automatic test system as claimed in claim 52, wherein

the testing means further includes a means for simultaneously testing a plurality of target applications at one location or at multiple locations.

82. (New) The automatic test system as claimed in claim 52, wherein

a fire-wall is either provided between the testing means and a network or between the communication network and the target applications or at both locations for access control.